

Александр Кирсанов, ВКонтакте



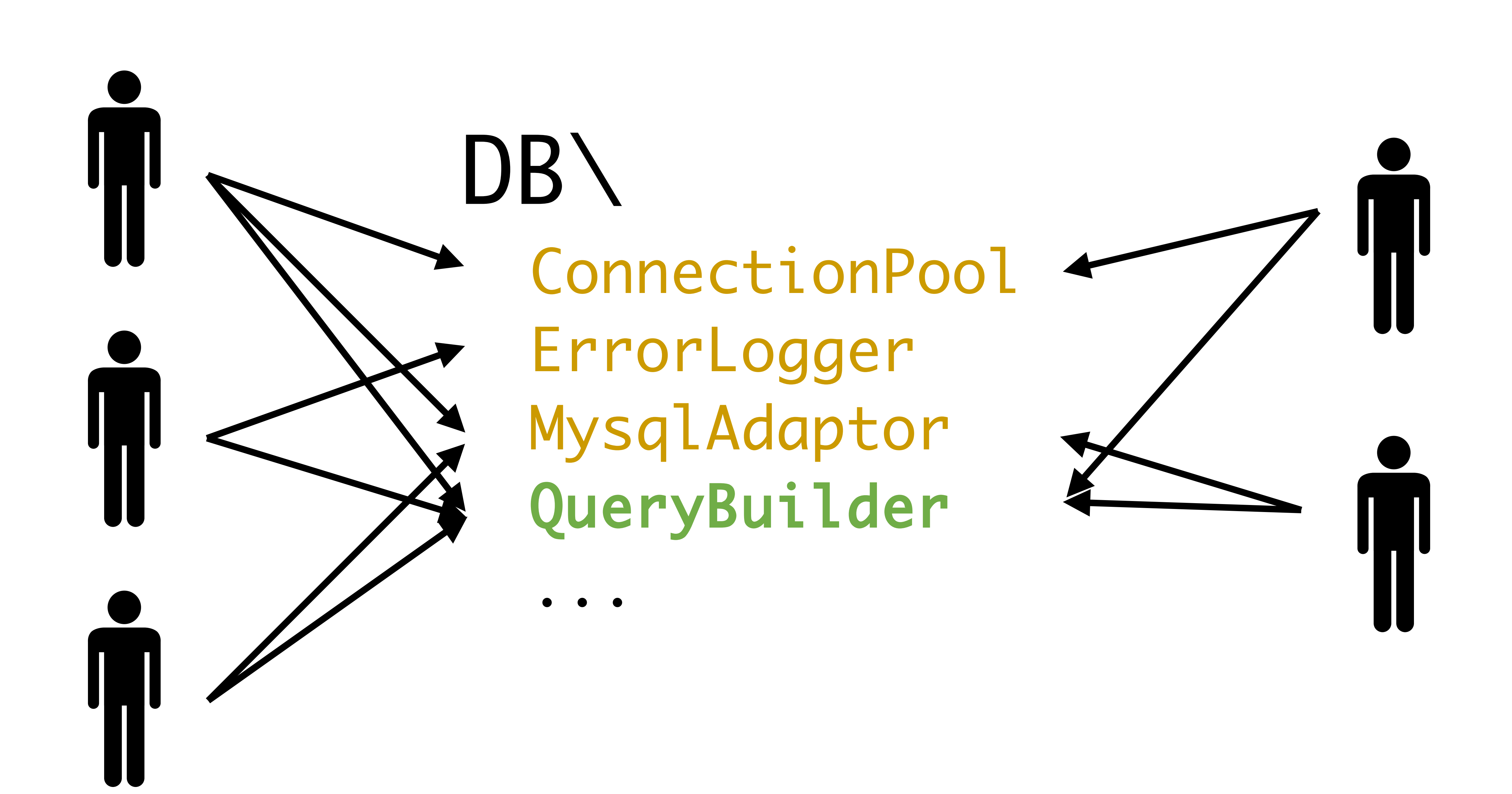


and p.name
if(parameters.contains(
hql +=
TypedQuery





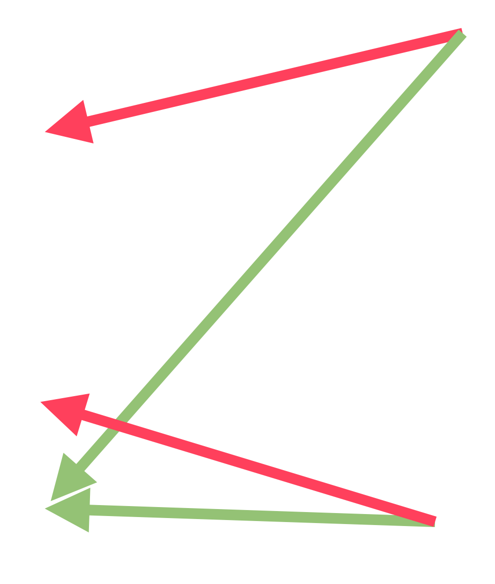
```
DB\
    ConnectionPool
    ErrorLogger
    MysqlAdaptor
    QueryBuilder
    ...
```

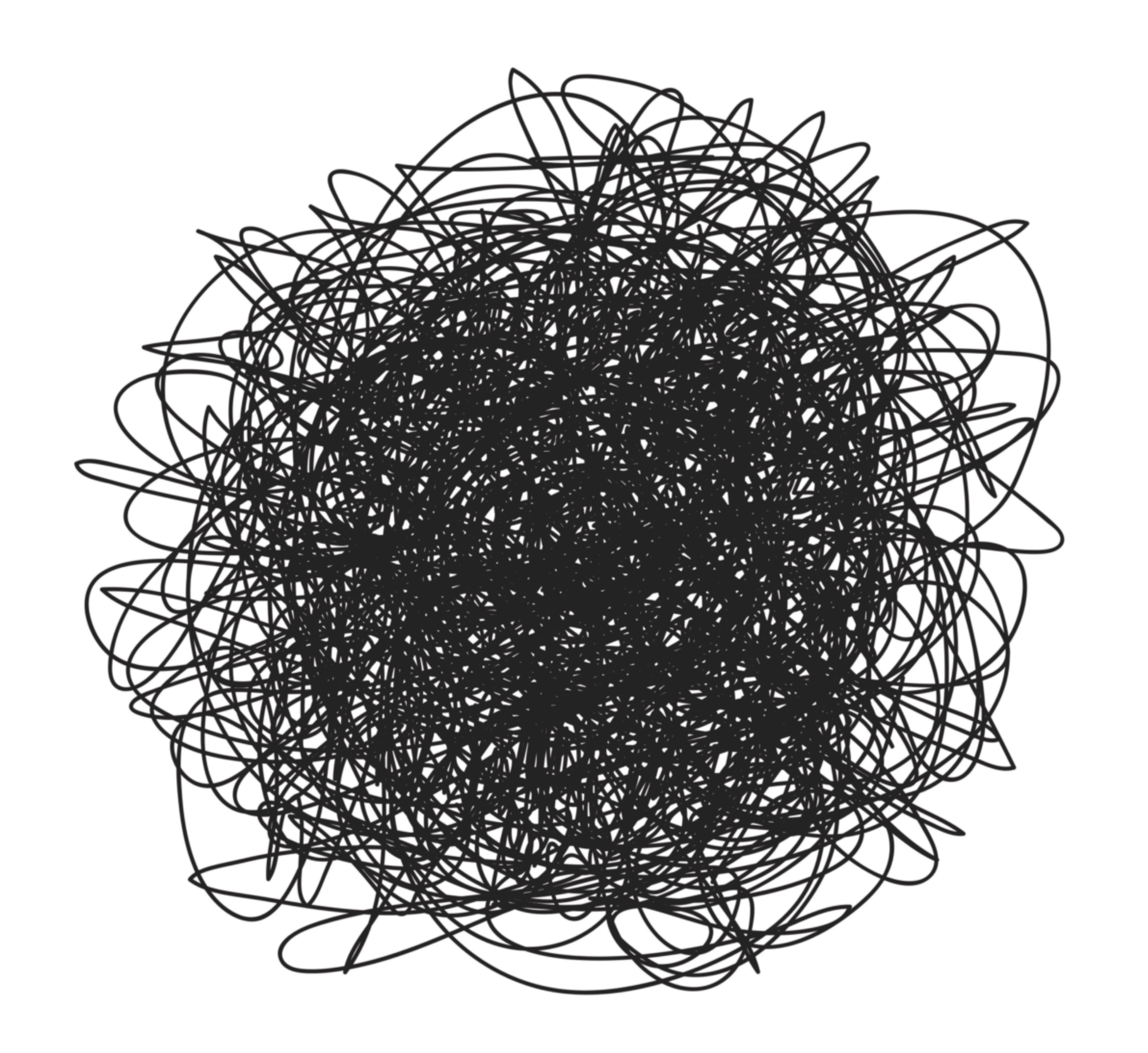

DB\
ConnectionPool
ErrorLogger
MysqlAdaptor
QueryBuilder
...

DB\



ConnectionPool
ErrorLogger
MysqlAdaptor
QueryBuilder
...





```
internal class MysqlAdaptor

private namespace DB\Impl
```

Go

Rust

modules

JS

CSS

Go

Rust

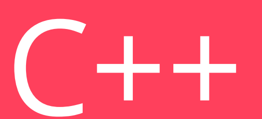
PHP

modules

JS

CSS

PHP **modules**



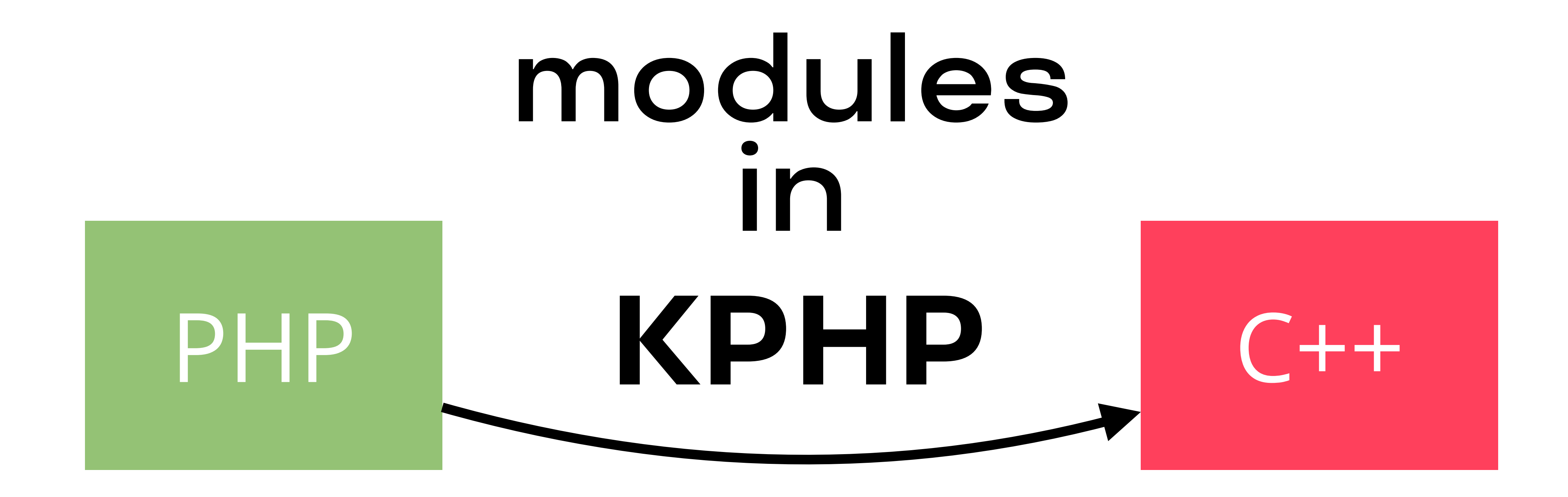
PHP



PHP

KPHP

C++


modules
in
KPHP
PHP
C++

modules
in



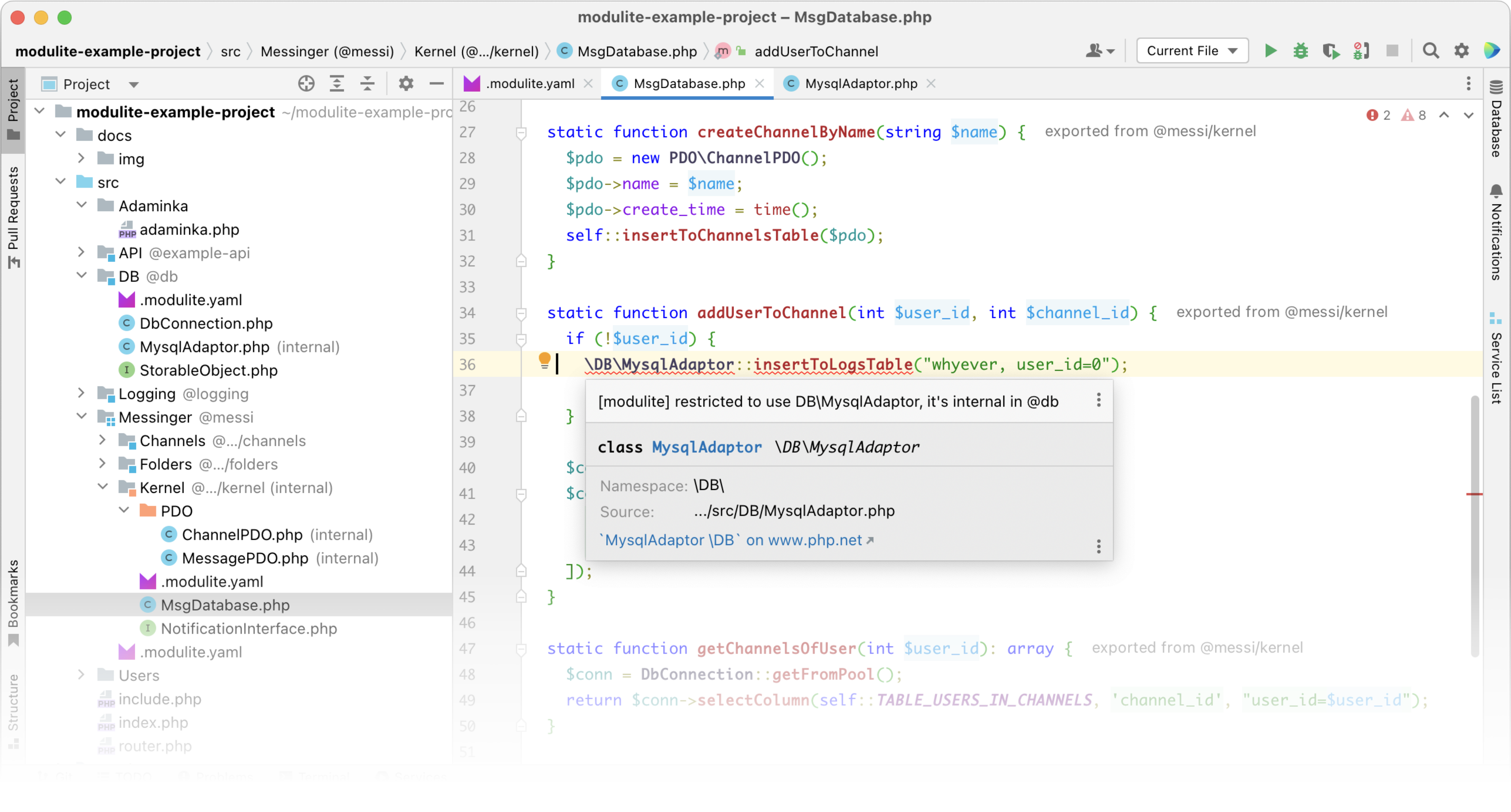
modules
in
PHP

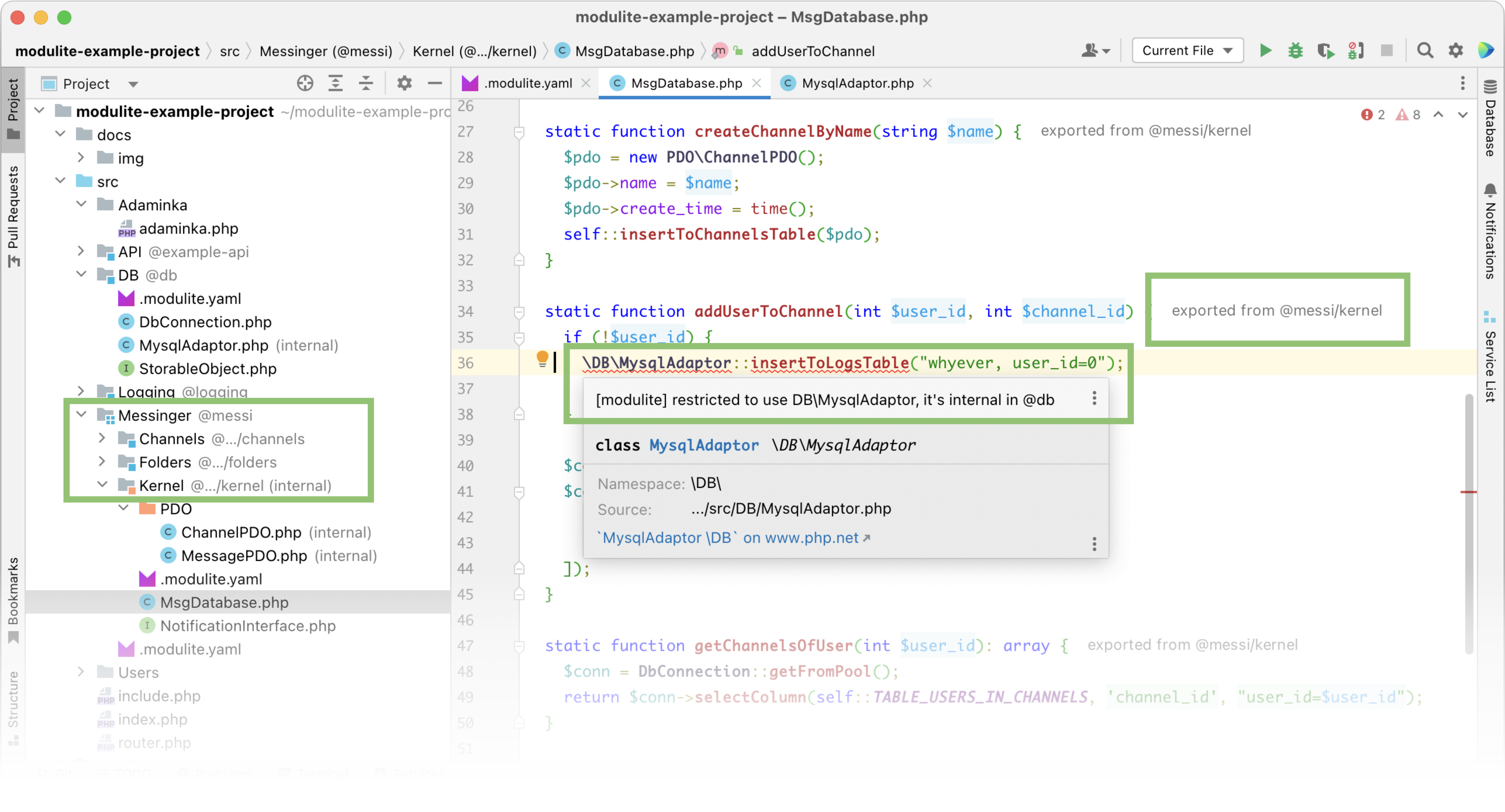
Modulite

Честная модульность внутри PHP

Установить







modulite-example-project – MsgDatabase.php
modulite-example-project
src
Messinger (@messi)
Kernel (@.../kernel)
MsgDatabase.php
addUserToChannel
Current File
Project
.modulite.yaml
MsgDatabase.php
MysqlAdaptor.php
modulite-example-project ~/modulite-example-pro
docs
img
src
Adaminka
adaminka.php
API @example-api
DB @db
.modulite.yaml
DbConnection.php
MysqlAdaptor.php (internal)
StorableObject.php
Logging @logging
Messinger @messi
Channels @.../channels
Folders @.../folders
Kernel @.../kernel (internal)
PDO
ChannelPDO.php (internal)
MessagePDO.php (internal)
.modulite.yaml
MsgDatabase.php
NotificationInterface.php
.modulite.yaml
Users
include.php
index.php
router.php
static function createChannelByName(string $name) {
exported from @messi/kernel
$pdo = new PDO\ChannelPDO();
$pdo->name = $name;
$pdo->create_time = time();
self::insertToChannelsTable($pdo);
}
static function addUserToChannel(int $user_id, int $channel_id)
exported from @messi/kernel
if (!$user_id) {
\DB\MysqlAdaptor::insertToLogsTable("whyever, user_id=0");
[modulite] restricted to use DB\MysqlAdaptor, it's internal in @db
class MysqlAdaptor \DB\MysqlAdaptor
Namespace: \DB\
Source: .../src/DB/MysqlAdaptor.php
`MysqlAdaptor \DB` on www.php.net
]);
}
static function getChannelsOfUser(int $user_id): array {
exported from @messi/kernel
$conn = DbConnection::getFromPool();
return $conn->selectColumn(self::TABLE_USERS_IN_CHANNELS, 'channel_id', "user_id=$user_id");
}
Pull Requests
Bookmarks
Structure
Database
Notifications
Service List

монолит

PHPStan

CI, Git

PHPStorm

Composer

internal

Modulite

вы

Modulite

+

internal

Месси







Месси



Адам



Messinger/

Месси



Messinger/

Адам



Adaminka/

Месси



Messinger/

Channels/

Notifications/

...

Folders/

Kernel/

```
Messinger
  Channels
    Notifications
      JoinNotification.php
      LeaveNotification.php
    ChannelActions.php
  Folders
```

```php
function adaminka_addUser(int $uid, int $channel_id) {
  MsgDatabase::addUserToChannel($uid, $channel_id);
  $n = new JoinNotification('you were added by admin');
  $n->send($uid);
  Logger::logSuccess("user $uid added to $channel_id");
}
```



- Messinger
  - Channels
    - Notifications
      - JoinNotification.php
      - LeaveNotification.php
    - ChannelActions.php
  - Folders

```php
function adaminka_addUser(int $uid, int $channel_id) {
    MsgDatabase::addUserToChannel($uid, $channel_id);
    $n = new JoinNotification('you were added by admin');
    $n->send($uid);
    Logger::logSuccess("user $uid added to $channel_id");
}
```



New Modulite

Name: @messi-channels

Description:

Folder: Channels

A folder to place the new modulite.

Namespace: \Messinger\Channels

Base namespace for the modulite.

Exported Symbols

- [-] Messinger\
  - [-] Channels\
    - [x] ChannelActions
    - [ ] Notifications\
      - [ ] JoinNotification
      - [ ] LeaveNotification

Selected symbols will be public, the rest will be internal and will not be available outside of the modulite.

Cancel OK

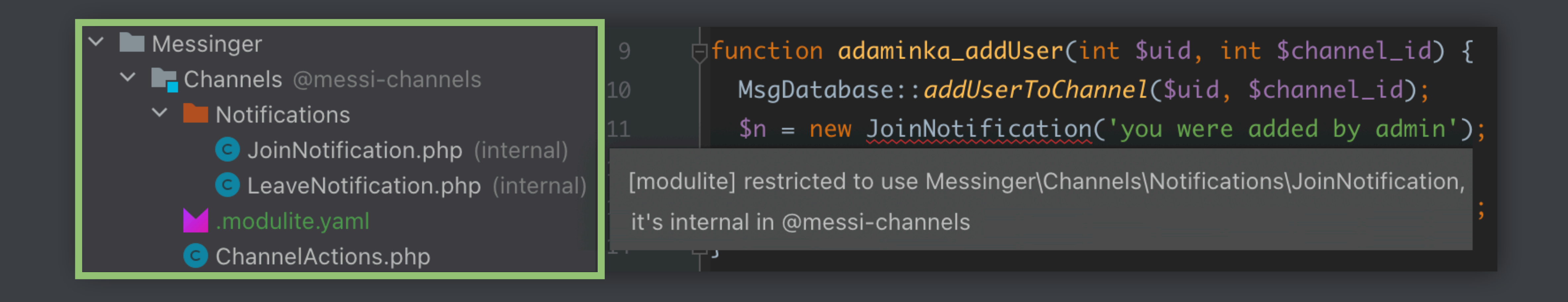


- Messinger
  - Channels @messi-channels
    - Notifications
      - JoinNotification.php (internal)
      - LeaveNotification.php (internal)
    - .modulite.yaml
    - ChannelActions.php

```php
function adaminka_addUser(int $uid, int $channel_id) {
  MsgDatabase::addUserToChannel($uid, $channel_id);
  $n = new JoinNotification('you were added by admin');
```

[modulite] restricted to use Messinger\Channels\Notifications\JoinNotification, it's internal in @messi-channels




Messinger
Channels @messi-channels
Notifications
JoinNotification.php (internal)
LeaveNotification.php (internal)
.modulite.yaml
ChannelActions.php
function adaminka_addUser(int $uid, int $channel_id) {
MsgDatabase::addUserToChannel($uid, $channel_id);
$n = new JoinNotification('you were added by admin');
[modulite] restricted to use Messinger\Channels\Notifications\JoinNotification,
it's internal in @messi-channels



Channels/.modulite.yaml

```yaml
name: "@messi-channels"
description: ""
namespace: "Messinger\\Channels\\"

# "Public API" of the modulite: classes, functions, constants, etc.
# Symbols not listed here will be internal.
1 total
export:
  1 class
  - "ChannelActions"
```



Modulite

+

require

Месси



`Messinger/`

Джун



`isUserSubscribed()`

```php
static function isUserSubscribed(int $channel_id): bool {   exported from @messi-channels
  $all_subs = MsgDatabase::getChannelsOfUser(currentUser()->id);
  return in_array($channe
  }
}
```

[modulite] restricted to call currentUser(), it's not required by @messi-channels

Add symbol to requires ⌥⇧↩   More actions... ⌥↩ ^. ⌥⇧F10



```
static function isUserSubscribed(int $channel_id): bool {   exported from @messi-channels
    $all_subs = MsgDatabase::getChannelsOfUser(currentUser()->id);
    return in_array($channel_id, $all_subs);
}
```

Add symbol to requires

Find usages in the current modulite



Channels/.modulite.yaml

```
# Dependencies: other modulites, global classes, defines, etc.
4 total, click to regenerate
require:
  1 class
  - "\\Messinger\\Kernel\\NotificationInterface"
  2 methods
  - "\\Messinger\\Kernel\\MsgDatabase::addUserToChannel()"
  - "\\Messinger\\Kernel\\MsgDatabase::getChannelsOfUser()"
  1 function
  - "\\currentUser()"
```



Channels/.modulite.yaml

```
# Dependencies: other modulites, global classes, defines, etc.
4 total, click to regenerate
require:
  1 class
  - "\\Messinger\\Kernel\\NotificationInterface"
  2 methods
  - "\\Messinger\\Kernel\\MsgDatabase::addUserToChannel()"
  - "\\Messinger\\Kernel\\MsgDatabase::getChannelsOfUser()"
  1 function
  - "\\currentUser()"
```



Итого. Модуль — это обычная папка с PHP-кодом.
С одной стороны, она определяет доступ "внутрь" через export.
С другой, она определяет доступ "наружу" через requires.

Modulite

+

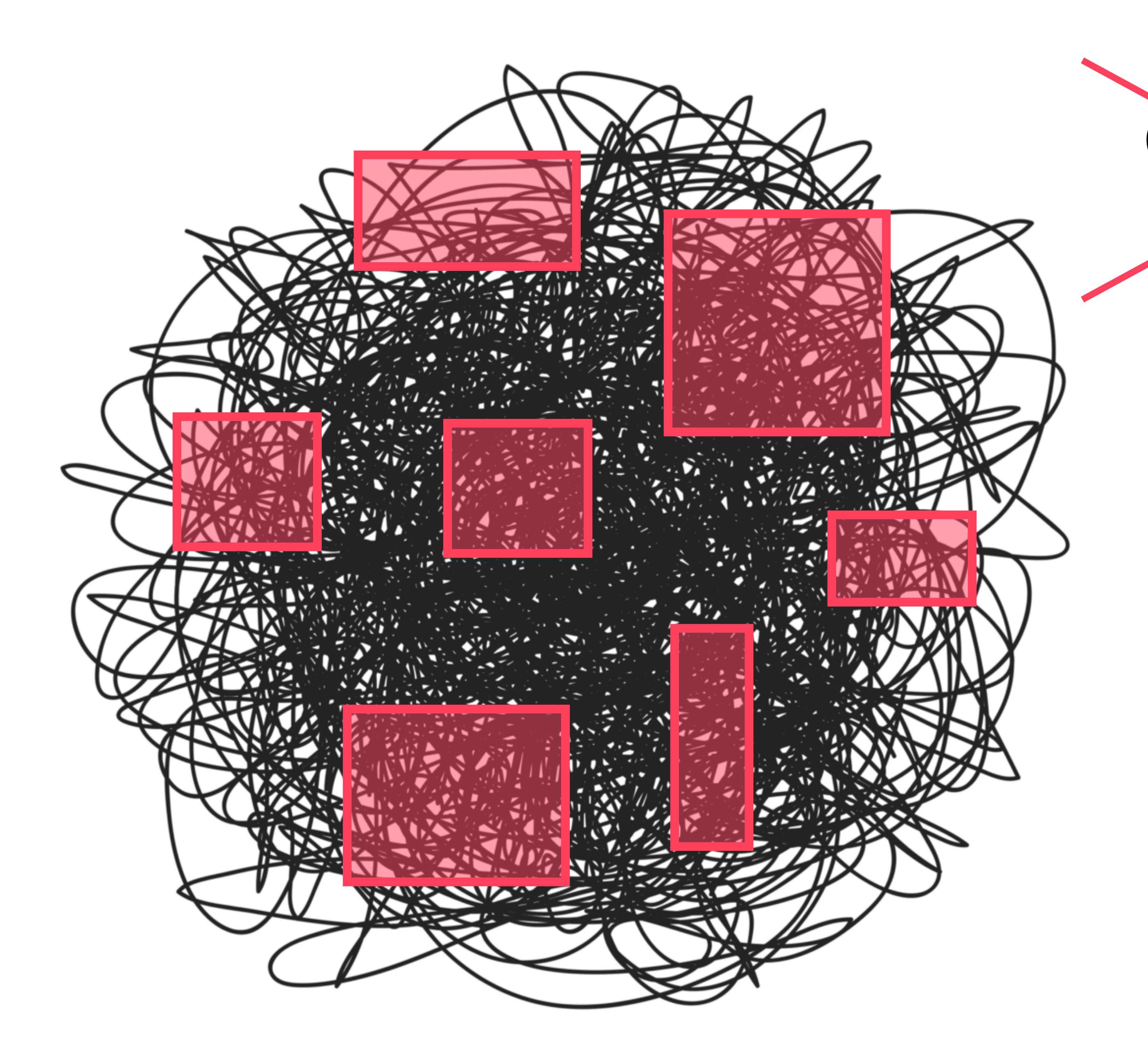
монолит

Цель

Не допустить неконтролируемого разрастания энтропии внутри монолита

Предпосылки

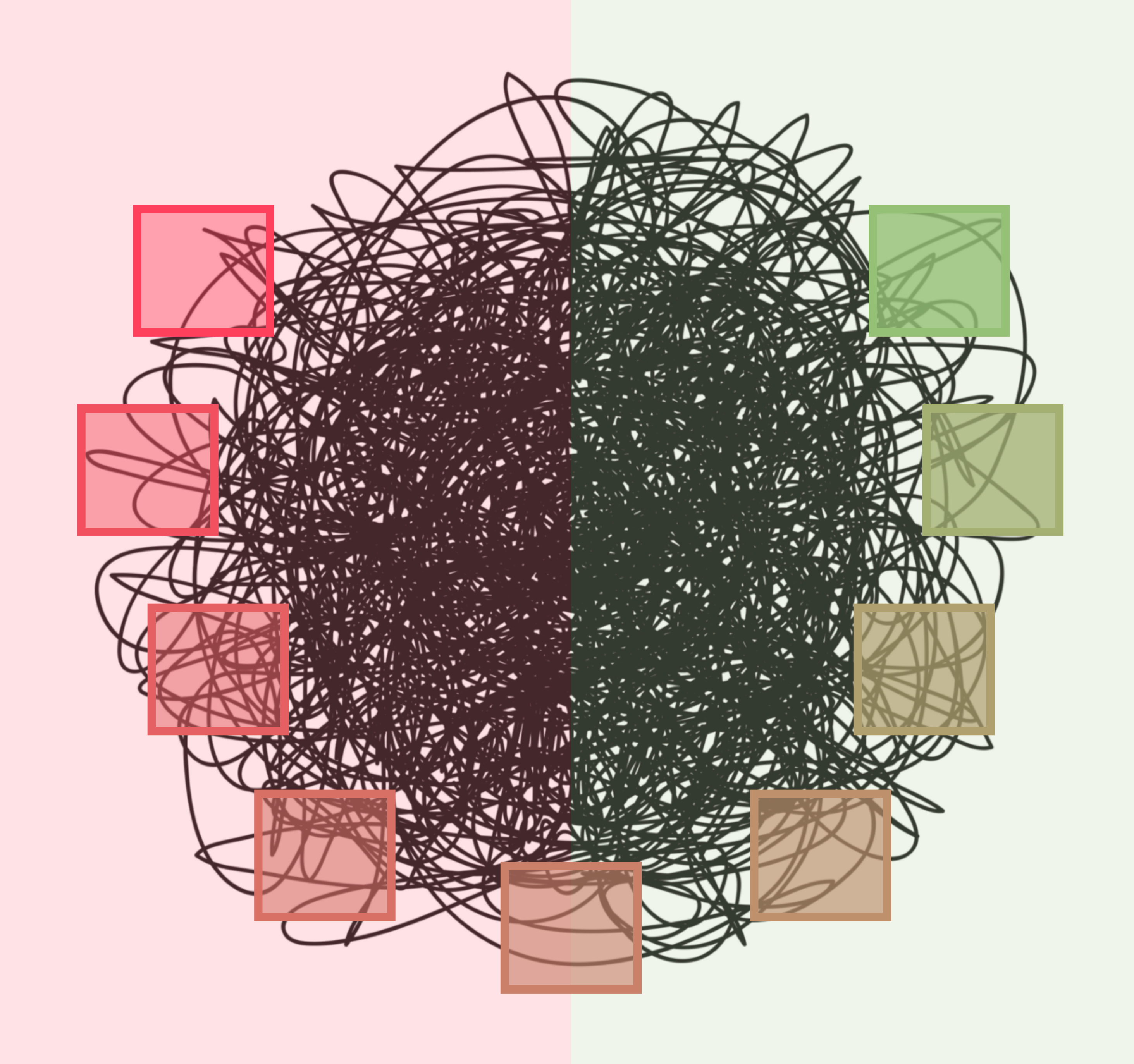
Изоляция отдельных папок **в существующем коде**



~~Composer package~~

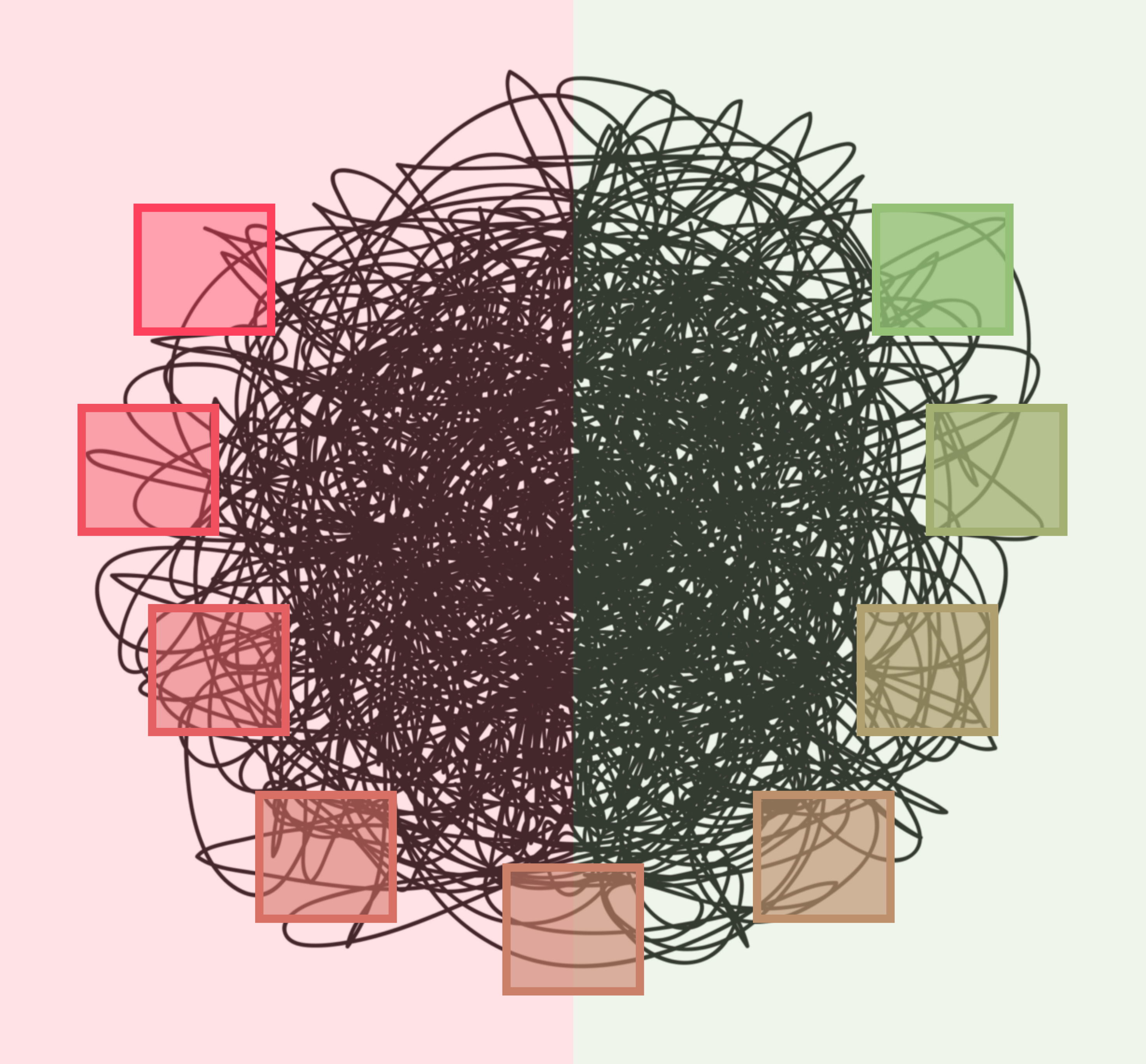
Модульность позволяет

Изолировать отдельные участки кода,
которые подчиняются правилам,
которые задаются владельцами этого участка



Module

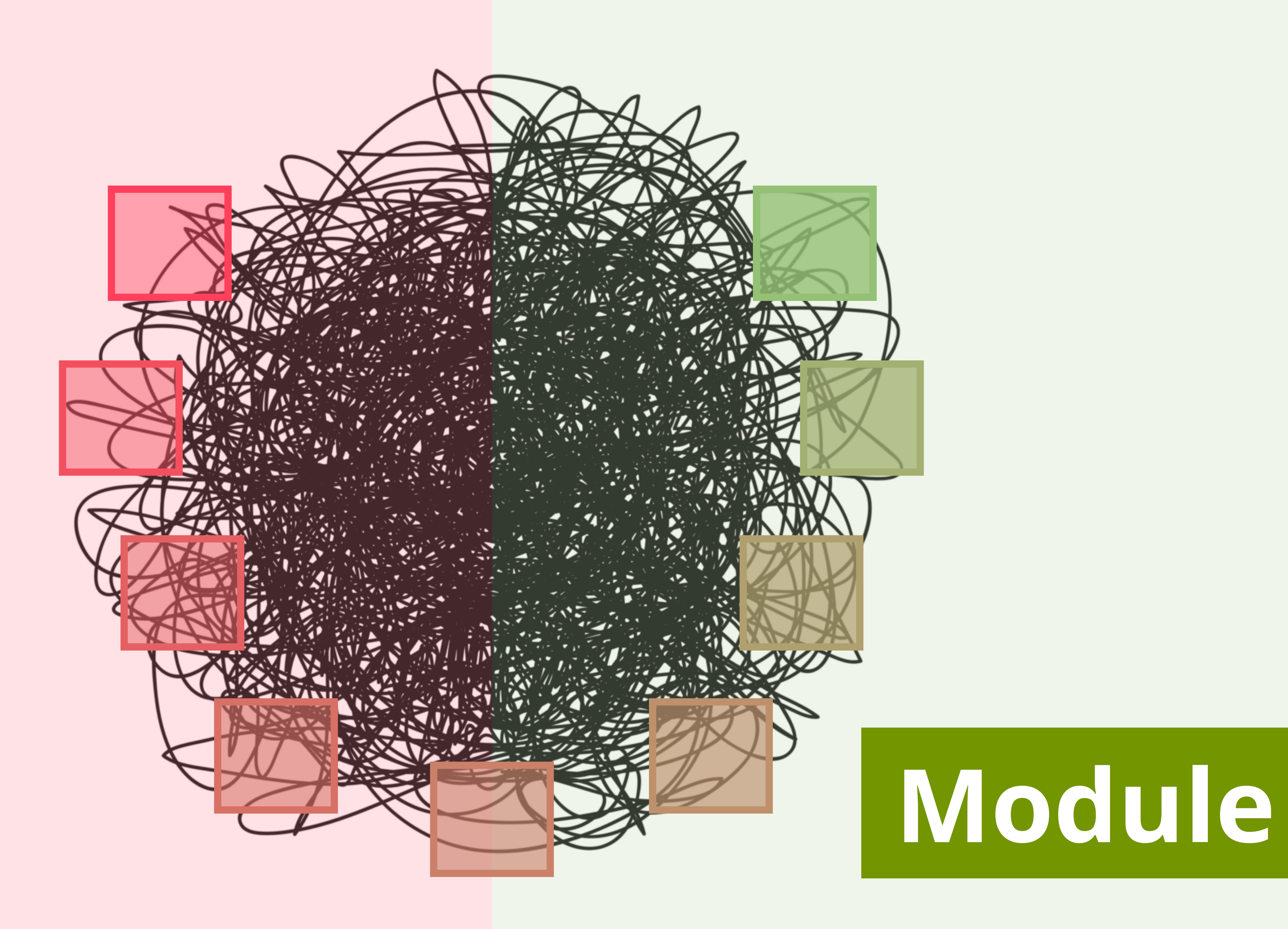
Composer package



Module

?

Module



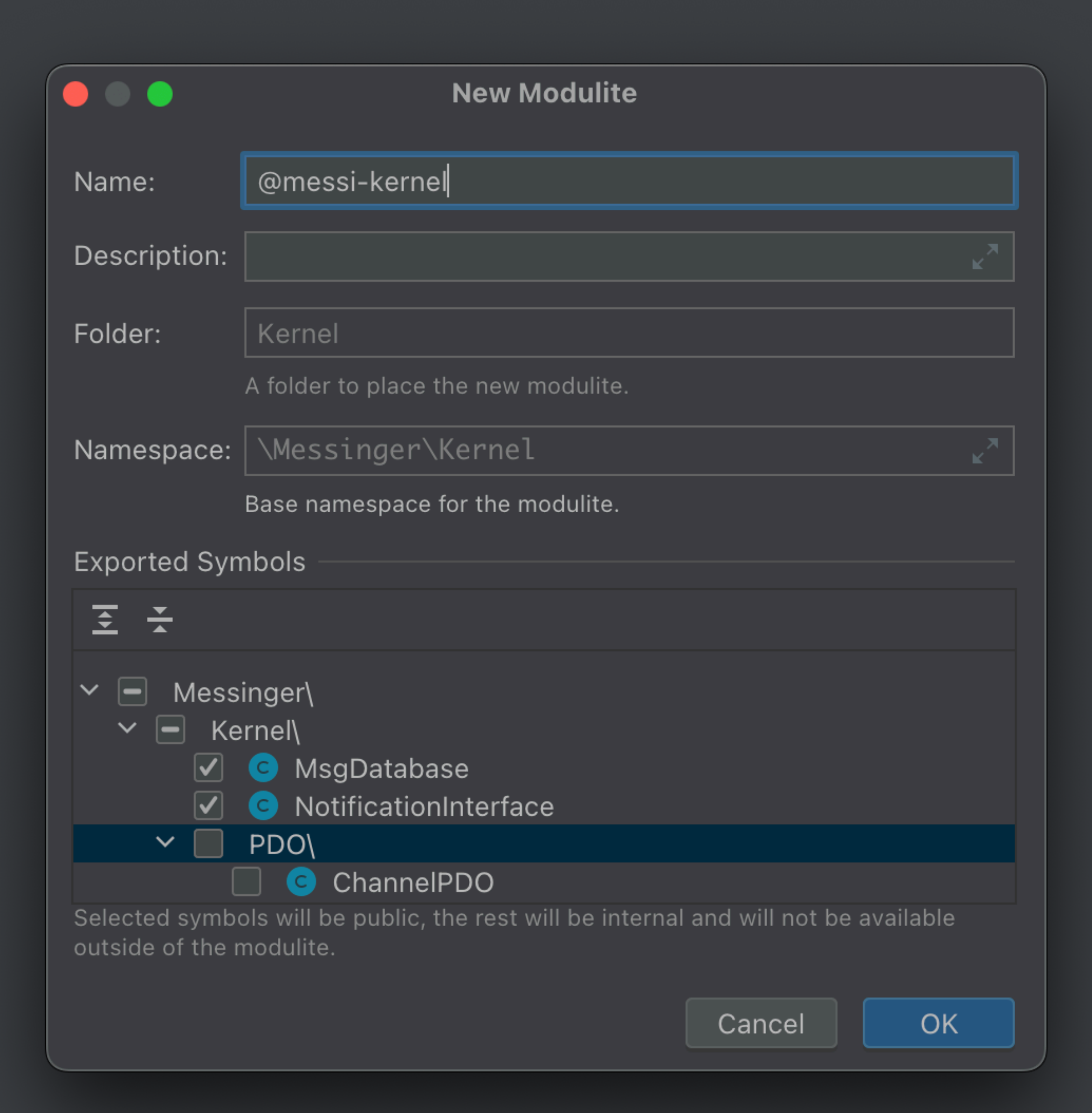
В общем, не найдя аналогов, мы сделали Modulite.
Мы скрестили наш опыт в IDE, чтобы это было удобным.
Наш опыт в разработке, чтобы это было правильным.
И наш опыт в компиляторах, чтобы это было быстрым.

Modulite

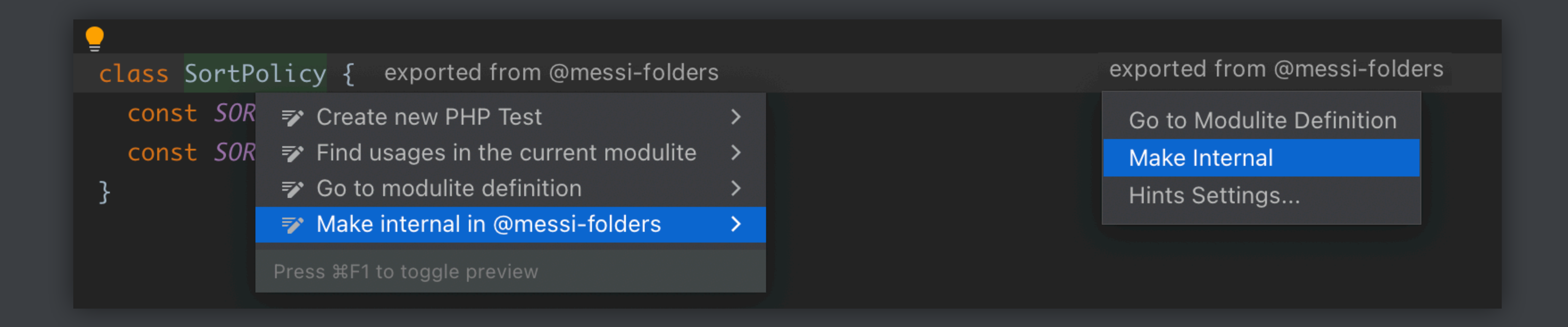
+

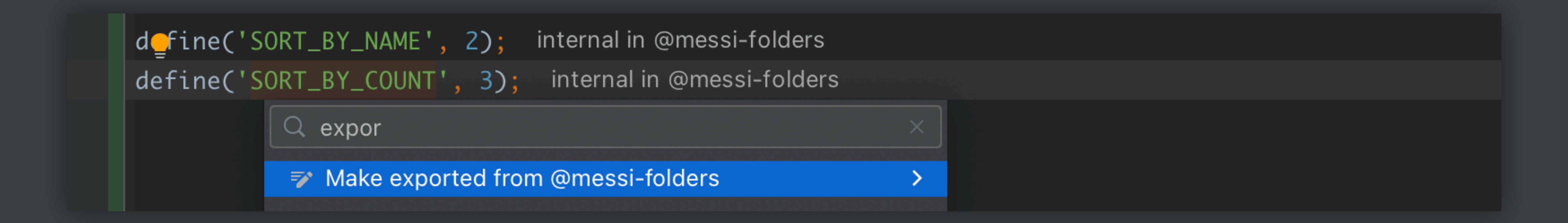
PHPStorm

Создание модуля из папки



Делаем символы internal и обратно





internal, если уже используется

```
class SortPolicy {   exported from @messi-folders
  const SORT
  const SORT
}
```

inte

Make internal in @messi-folders

```
// in API
$policy = SortPolicy::SORT_BY_NAME;
$folders = SortImplementation::sortFolders($folders, $policy);
```

перестанет компилироваться

internal, если уже используется

```
class SortPolicy {   internal in @messi-folders (visible for \API\ApiMessinger::getMyFolders())
  const SORT_BY_NAME         = 0;
  const SORT_BY_UNREAD_COUNT = 1;
}


// in API
$policy = SortPolicy::SORT_BY_NAME;
$folders = SortImplementation::sortFolders($folders, $policy);
```

```
# Granting partial access to internal symbols, "as an exception".
allow-internal-access:
  "\\API\\ApiMessinger::getMyFolders()":
    - "SortPolicy"
```

Скрываем члены публичного класса

```
static function isUserSubscribed(int $channel_id): bool {   exported from @messi-channels
  $all_subs = MsgDatabase::
  return in_array($channel_
}
```

inter

Make internal in @messi-channels

Скрываем члены публичного класса

```
static function isUserSubscribed(int $channel_id): bool {   exported from @messi-channels
  $all_subs = MsgDatabase::
  return in_array($channel_
}
```

inter

Make internal in @messi-channels

```
# Class members to exclude, they override "export".
force-internal:
  - "ChannelActions::isUserSubscribed()"
```

Новый код и require

```
function send(int $user_id) {
  global $parsed_session;
}
```

[modulite] restricted to use global $parsed_session, it's not required by @messi-channels

Add symbol to requires ⌥⇧↩ More actions... ⌥↩ ^. ⌥⇧F10

Новый код и require

```
# Dependencies: other modulites, global classes, defines, etc.
7 total, click to regenerate
require:
  2 modulites
  - "@api"
  - "@users"
  1 class
  - "\\Messinger\\Kernel\\NotificationInterface"
  2 methods
  - "\\Messinger\\Kernel\\MsgDatabase::addUserToChannel()"
  - "\\Messinger\\Kernel\\MsgDatabase::getChannelsOfUser()"
  1 function
  - "\\currentUser()"
  1 global variable
  - "$global_var"
```

Новый код и require

```
# Dependencies: other modulites, global classes, defines, etc.
7 total, click to regenerate
require:
  2 modulites
  - "@api"
  - "@users"
  1 class
  - "\\Messinger\\Kernel\\NotificationInterface"
  2 methods
  - "\\Messinger\\Kernel\\MsgDatabase::addUserToChannel()"
  - "\\Messinger\\Kernel\\MsgDatabase::getChannelsOfUser()"
  1 function
  - "\\currentUser()"
  1 global variable
  - "$global_var"
```

Новый код и require

```
# Dependencies: other modulites, global classes, defines, etc.
5 total, click to regenerate
require:
  3 modulites
  - "@api"
  - "@messi-kernel"
  - "@users"
  1 function
  - "\\currentUser()"
  1 global variable
  - "$global_var"
```

Новый код и require

```
1 class
- "\\Messinger\
2 methods
- "\\Messinger\\Kernel\\MsgDatabase::addUserToChannel()"
- "\\Messinger\\Kernel\\MsgDatabase::getChannelsOfUser()"
```

MsgDatabase::**addUserToChannel**(*int* **$user_id**, *int* **$channel_id**): *void*

Вложенные модули, подмодули

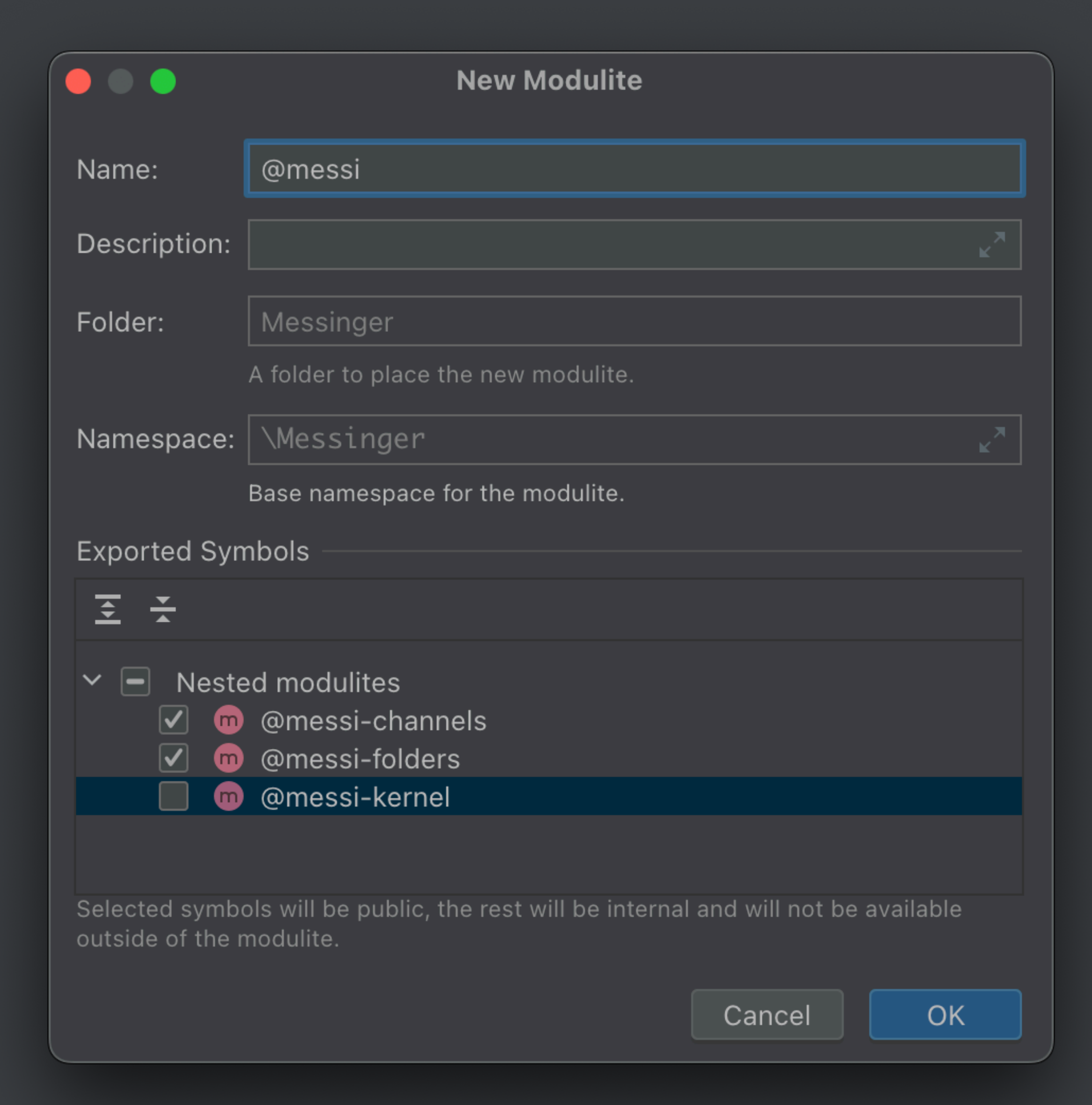
```
Messinger/
  Channels/    @messi-channels
    ...
  Folders/     @messi-folders
    ...
  Kernel/      @messi-kernel
    ...
```

Вложенные модули, подмодули

```
Messinger/
  Channels/     @messi-channels
    ...
  Folders/      @messi-folders
    ...
  Kernel/       @messi-kernel
    ...
```

```
Messinger/      @messi
  Channels/     @messi/channels
    ...
  Folders/      @messi/folders
    ...
  Kernel/       @messi/kernel (internal)
    ...
```

Вложенные модули, подмодули

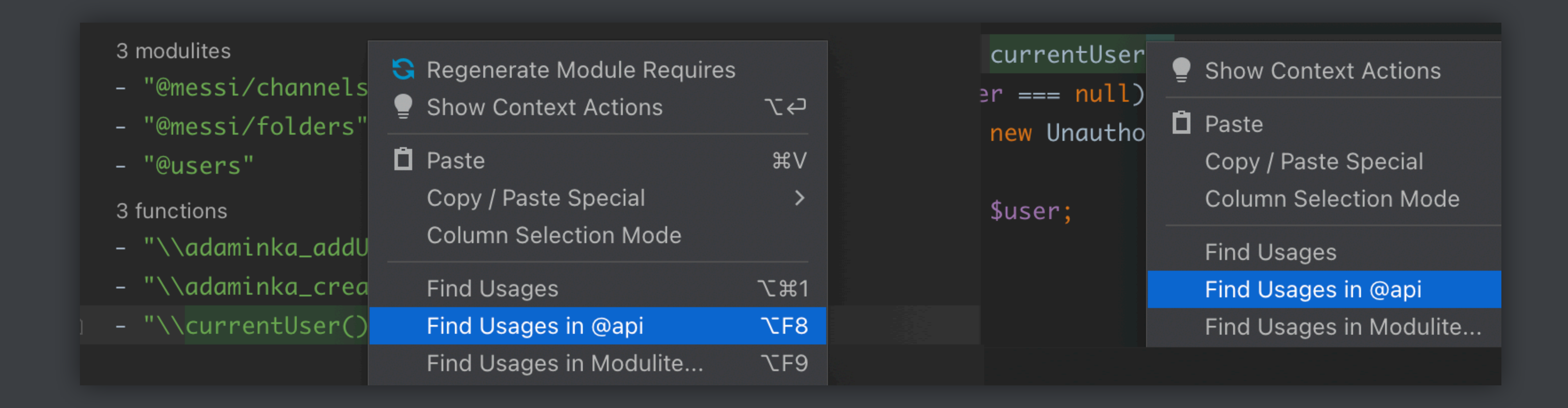


Вложенные модули, подмодули

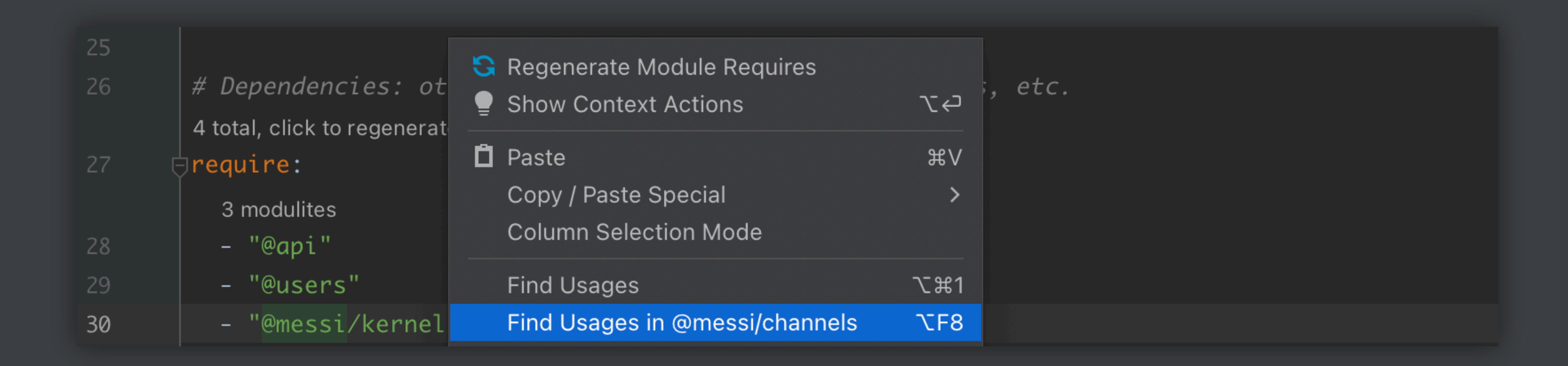
```
7 total, click to regenerate
require:
  4 modulites
  - "@messi/channels"
  - "@messi/folders"
  - "@messi/kernel"
  - "
  3 functions
```

[modulite] restricted to use @messi/kernel, it's internal in @messi

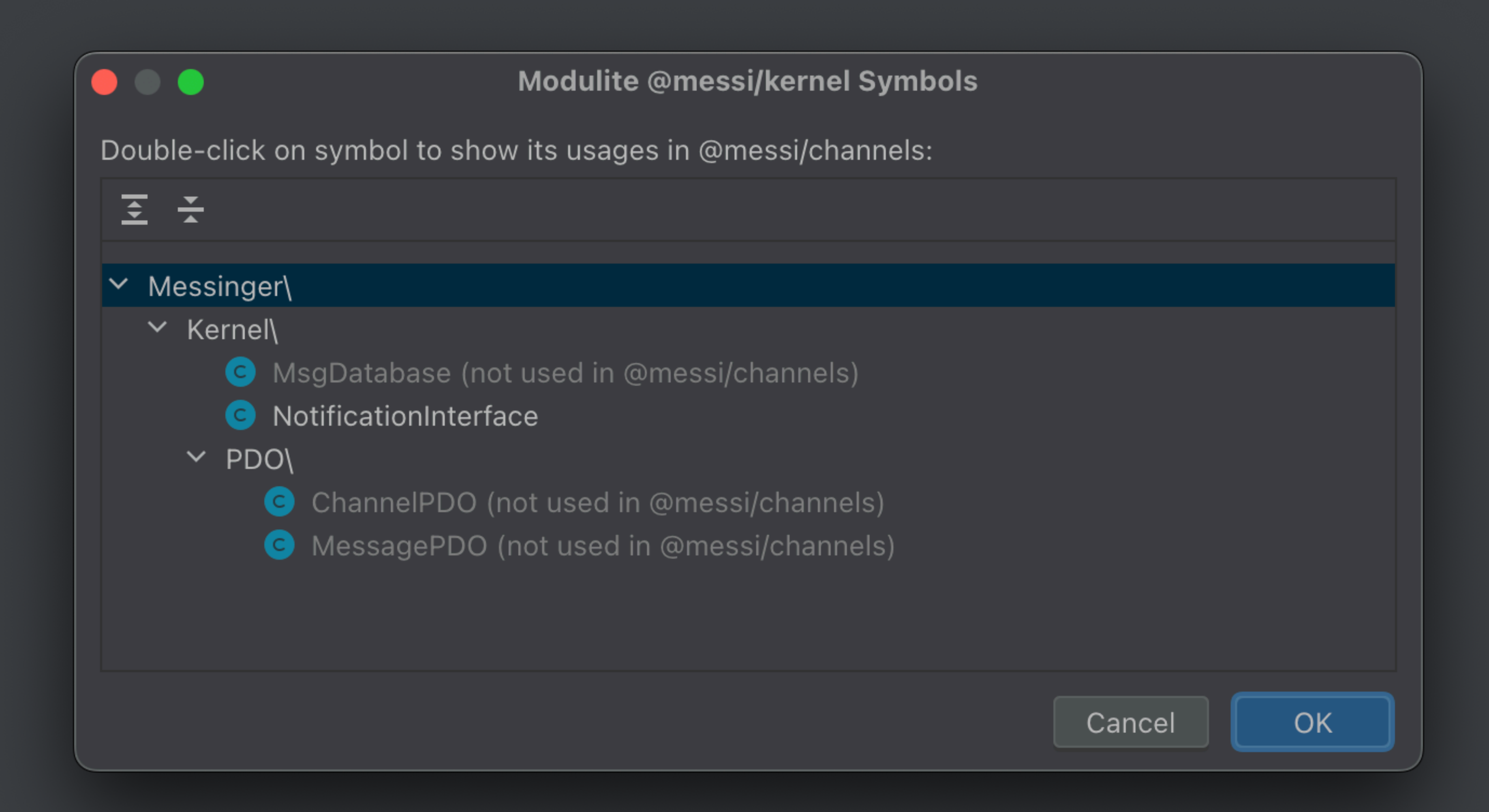
Find usages внутри модуля



Зависимость от чужого модуля



Зависимость от чужого модуля



Имена модулей в интерфейсе

```
static function createZero(): self {   exported from @.../pdo
  return new self;
}
```

```
static function createZero(): self {   exported from @messinger/kernel/pdo
  return new self;
}
```

Имена модулей в интерфейсе

monolit 〉 Messinger (@messinger) 〉 Kernel (@.../kernel) 〉 PDO (@.../pdo) 〉 MessagePDO.php 〉 MessagePDO

- Messinger @messinger
  - Channels @.../channels
    - Notifications
      - JoinNotification.php (internal)
      - LeaveNotification.php (internal)
    - .modulite.yaml
    - ChannelActions.php
  - Folders @.../folders
  - Kernel @.../kernel
    - PDO @.../pdo (internal)
      - .modulite.yaml
      - ChannelPDO.php

Плагин — это удобный UI над конфигом .modulite.yaml.
Именно файлик хранится под гитом, именно изменения в нём видны на ревью при добавлении зависимостей или исключений.

```yaml
name: "@modulite-name"
description: "..."
namespace: "Some\\Namespace\\"

export:
  - "ClassInNamespace"
  - "OrConcrete::CLASS_MEMBER"
  and others

force-internal:
  - "ClassInNamespace::staticMethod()"
  and others

require:
  - "@another-modulite"
  - "#composer/package"
  - "\\GlobalClass"
  and others

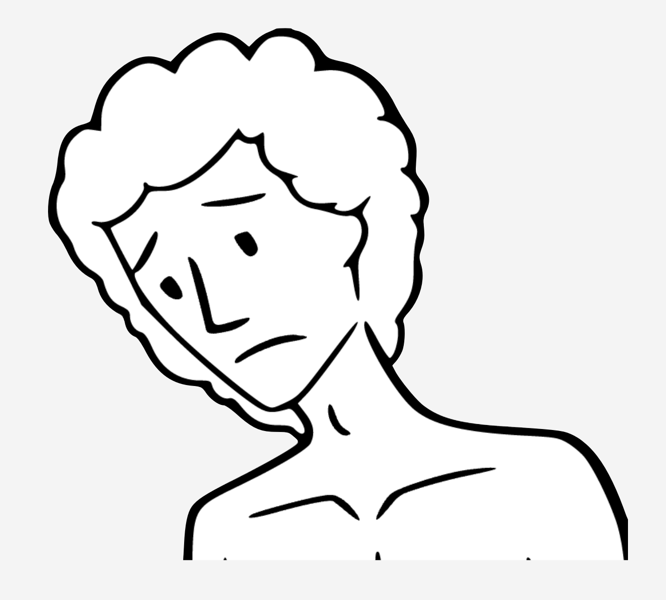
allow-internal-access:
  "@rpc":
    - "ClassAllowedForRpcModule"
    - "OrConcrete::method()"
  and others
```

Modulite

+

исключения

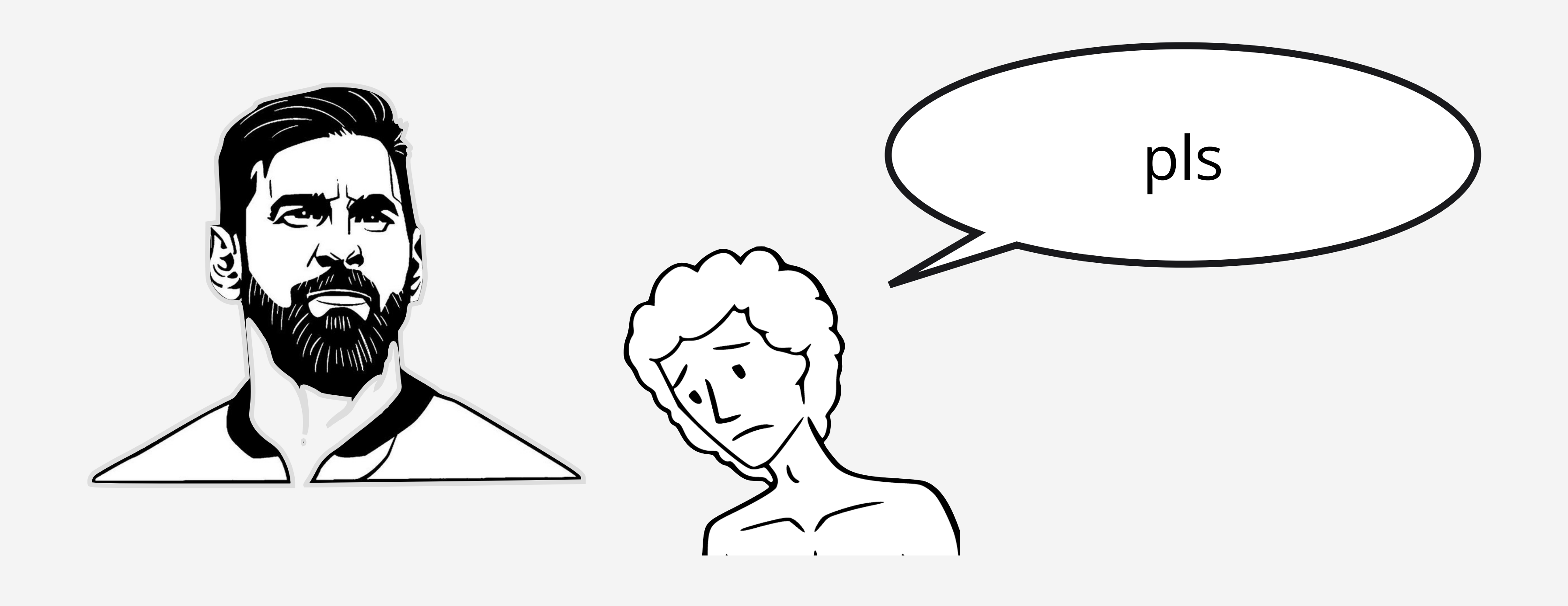


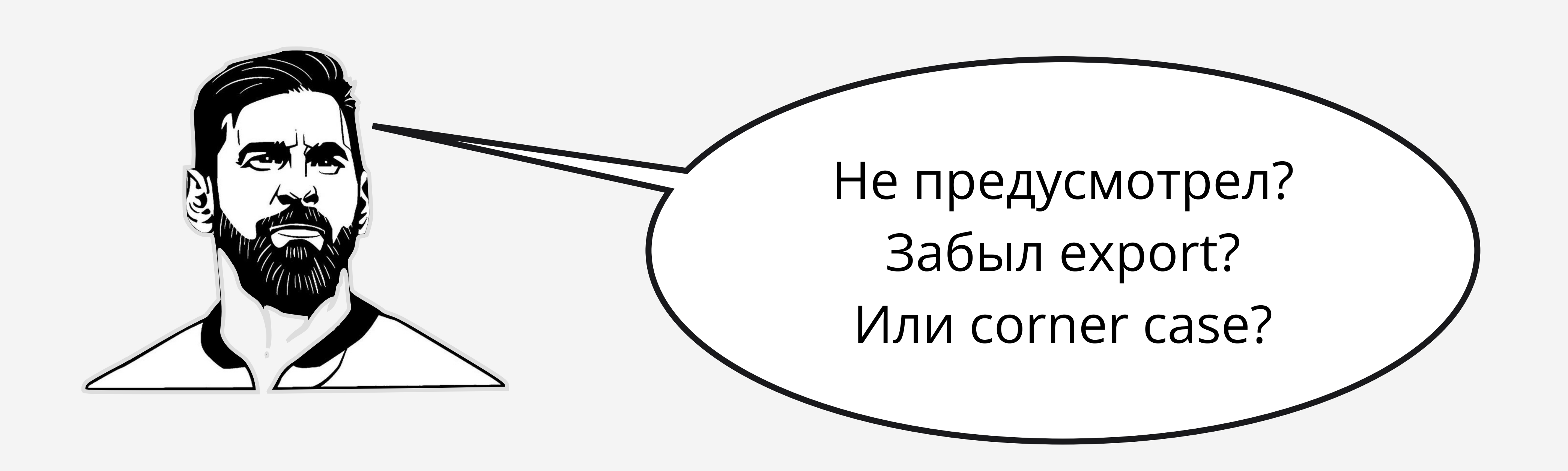


@messi-kernel

adaminka_addUser()

MsgDatabase (internal)


pls

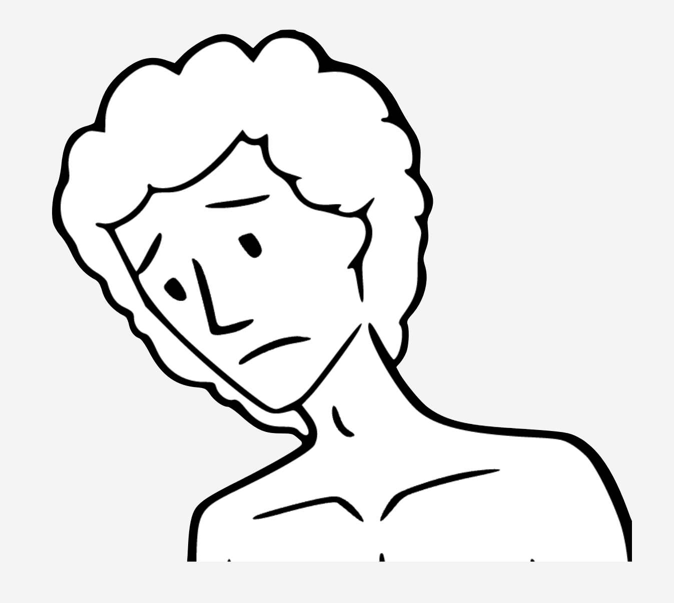

Не предусмотрел?
Забыл export?
Или corner case?

Kernel/.modulite.yaml

```yaml
allow-internal-access:
  "\\adaminka_addUser()":
    - "MsgDatabase::insertUser()"
    - "MsgDatabase::TABLE_MESSAGES"
```







@messi-kernel

adaminka_addUser()

MsgDatabase (internal)

Именно благодаря конфигу работают проверки
во время компиляции, в Git-хуках и в Teamcity:
Поскольку есть инструменты, анализирующие конфиг.

Modulite

+

PHPStan

Modulite + PHPStan

Сделали плагин для Open Source сообщества

Парсит yaml-файлы, валидирует, резолвит через рефлексию

Проверяет модульность по тем же самым правилам

Modulite + PHPStan

Не удалось пока победить PHPStan-кэш, т.к. php-файлы не меняются

Если ошибки выглядят старыми, сброс кэша помогает всегда

Modulite + PHPStan

```
------ -----------------------------------------------------------------------------------------------------------------------
 Line   Adaminka/adaminka.php
------ -----------------------------------------------------------------------------------------------------------------------
 9      [modulite] restricted to call Messinger\Kernel\MsgDatabase::addUserToChannel(), @messi/kernel is internal in @messi
 10     [modulite] restricted to use Messinger\Channels\Notifications\JoinChannelNotification, it's internal in @messi/channels
------ -----------------------------------------------------------------------------------------------------------------------

------ ---------------------------------------------------------------------------------------------
 Line   Messinger/Channels/ChannelActions.php
------ ---------------------------------------------------------------------------------------------
 14     [modulite] restricted to use Messinger\Kernel\PDO\ChannelPDO, it's internal in @messi/kernel
------ ---------------------------------------------------------------------------------------------

------ -----------------------------------------------------------------------------------------
 Line   Messinger/Kernel/MsgDatabase.php
------ -----------------------------------------------------------------------------------------
 36     [modulite] restricted to call DB\MysqlAdaptor::insertToLogsTable(), it's internal in @db
------ -----------------------------------------------------------------------------------------
```

Modulite

+

KPHP

Modulite + KPHP

Изначальная, эталонная версия, встроена в KPHP

Нет проблем с кэшем, т.к. KPHP анализирует проект целиком

Используется в VKCOM

Modulite + KPHP

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Compilation error at stage: Check func calls and vararg, gen by modulite-check-rules.cpp:327
  Adaminka/adaminka.php:9  in adaminka_addUser
    MsgDatabase::addUserToChannel($user_id, $channel_id);

[modulite] restricted to call Messinger\Kernel\MsgDatabase::addUserToChannel(), @messi/kernel is internal in @messi



~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Compilation error at stage: Check func calls and vararg, gen by modulite-check-rules.cpp:318
  Messinger/Kernel/MsgDatabase.php:36  in MsgDatabase::addUserToChannel
    \DB\MysqlAdaptor::insertToLogsTable("whyever, user_id=0");

[modulite] restricted to call DB\MysqlAdaptor::insertToLogsTable(), it's internal in @db
```

KPHP

В KPHP встроена полная поддержка Modulite. Он читает yaml-файлики и проверяет код на все правила.

noverify

В наш линтер Modulite не встроен, т.к. это несовместимо с diff-режимом. Да и не нужно: у нас везде KPHP.

PHPStan

Для сообщества мы сделали PHPStan-плагин. Это позволяет использовать Modulite и в обычных PHP-проектах.

CI и git hooks

На препуше гоняется KPHP, так что пройдут все проверки. А если пушнуть в обход, то при сборке уж точно свалится.

PHPStorm

В PHPStorm все проверки выполняются этим плагином. Он автоматически генерит конфиги и показывает ошибки.

Другие IDE

Для других IDE плагинов нет. Я не понимаю, как можно разрабатывать большие проекты, сидя в vim.

Modulite удобно использовать не только в монолите.
А например, разрабатывая любой Composer-пакет:
почему бы не писать его внутренности модульными?

Modulite

+

Composer

Герасим



Герасим



voice-text

Герасим



```
voice-text
impl/   @impl
   EmojiTable.php
   ...
TextToSpeech.php
Transliteration.php
```

Composer-пакет — обычный проект

```
src
  impl @impl
    .modulite.yaml
    EmojiTable.php (internal)
    Helpers.php
  TextToSpeech.php
  Transliteration.php
  WaveMultiplier.php
vendor
composer.json
```

```php
class Helpers {   exported from @impl
  static function stripEmoji(string $s) {   exported from @impl
    $symbols = preg_split('//u', $s);
    $not_emoji = array_filter($symbols,
      fn($c) => !in_array($c, EmojiTable::NATIVE_EMOJI_LIST)
    );
    return implode('', $not_emoji);
  }
```



Месси



Герасим



AudioRecognition

↓

vk/voice-text

Подключение в монолит

```
2 total, click to regenerate
require:
  1 modulite
  - "@messi/kernel"
  1 composer package
  - "#vk/voice-text"
```



Подключение в монолит

```
2 total, click to regenerate
require:
  1 modulite
  - "@messi/kernel"
  1 composer package
  - "#vk/voice-text"
```

Неявный модуль

`#vk/voice-text`

`#vk/voice-text/@impl`



Проверки работают автоматически

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Compilation error at stage: Inline defines pass, gen by modulite-check-rules.cpp:318
  Messinger/Channels/ChannelActions.php:31  in ChannelActions::leave
    \VK\VoiceText\impl\EmojiTable::NATIVE_EMOJI_LIST;

[modulite] restricted to use VK\VoiceText\impl\EmojiTable, it's internal in #vk/voice-text/@impl
```



Явные export
у Composer-пакета





#vk/voice-text

adaminka_addUser()

class Transliteration

{projectRoot}/.modulite.yaml

```yaml
name: "<composer_root>"
namespace: "VK\\VoiceText\\"

export:
  - "TextToSpeech"
  - "WaveMultiplier"
  # - "@impl"

force-internal:
  # тоже работает

require:
  # оставляем пустым, генерится из composer.json
```



.modulite.yaml + composer.json

- src
  - impl @impl
  - TextToSpeech.php
  - Transliteration.php (internal)
  - WaveMultiplier.php
- vendor
- .modulite.yaml

```
class Transliteration {   internal in <composer_root>
  static function rusToTranslit(string $s) {   internal in <composer
    if ($s === '')
      return '';
    return impl\Helpers::translitConsideringSounds($s);
  }
```







#vk/voice-text

adaminka_addUser()

class Transliteration

(internal)

Composer-пакет с точки зрения Modulite
ничем не отличается от обычного проекта.
А внутри монолита пакет становится неявным модулем.

Modulite

+

этимология

модуль

\+

монолит

модуль

+

монолит

модулит

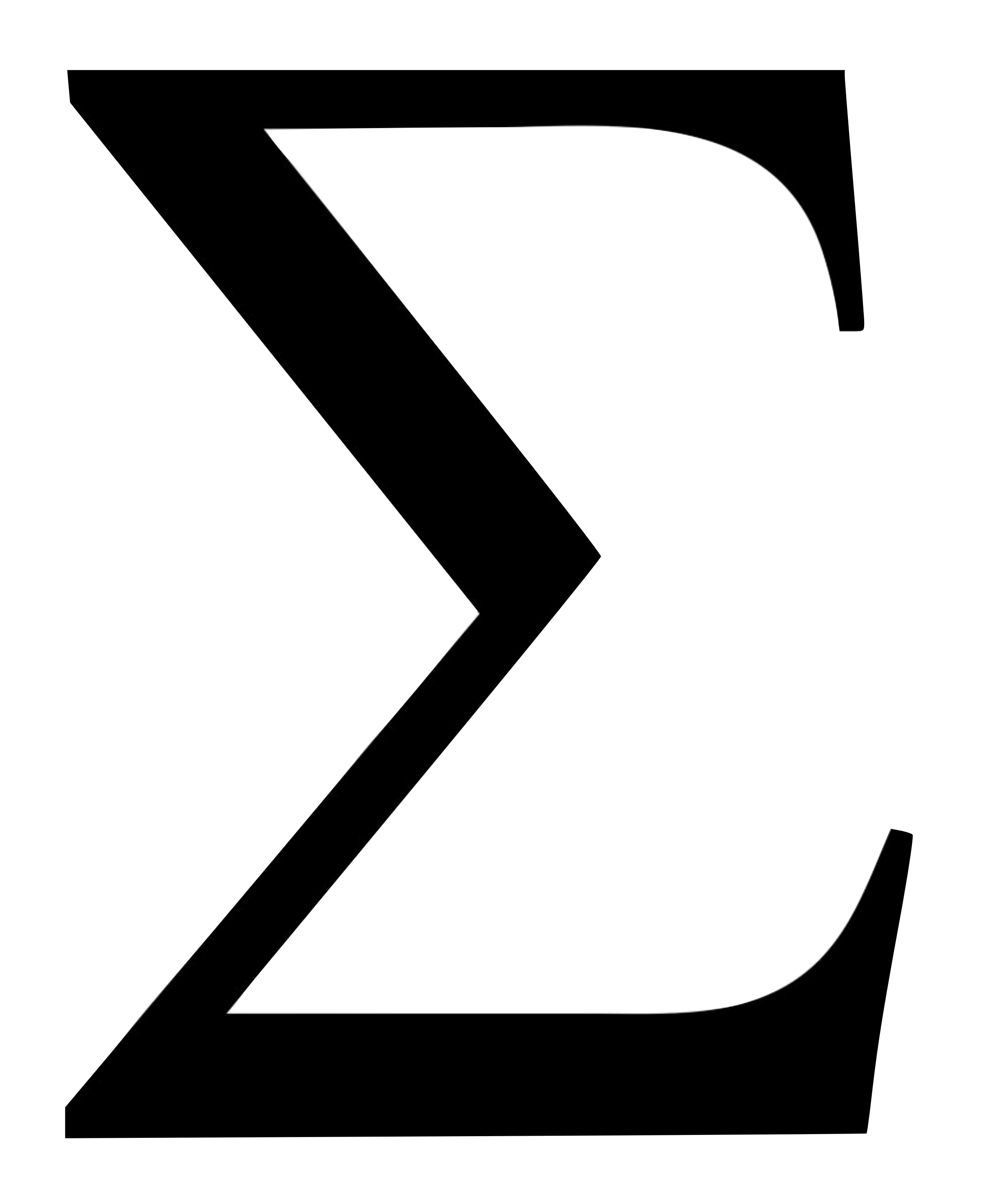
модуль
+
монолит

модулит = modulite

модуль
+
монолит

модулит = modulite

module
+
lite



Подводя итоги, Modulite дополняет язык PHP модулями,
не вмешиваясь в его синтаксис, а удобно находясь рядом.
Он подходит как для старта новых проектов, так и позволяет
зафиксировать состояние монолита и рефакторить итеративно.

Modulite



вы

https://github.com/VKCOM/modulite

Обратная связь и комментарии к докладу по ссылке

Александр Кирсанов,
руководитель команды KPHP



